1-1-2002

VRSpatial: a program for designing spatial 4C mechanisms in virtual reality

Denis Vitalievich Dorozhkin
Iowa State University

VRSpatial: A program for designing spatial 4C mechanisms in virtual reality

by

Denis Vitalievich Dorozhkin

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Mechanical Engineering

Program of Study Committee:
Judy M. Vance (Major Professor)
Donald R. Flugrad
Thomas J. Rudolphi

Iowa State University

Ames, Iowa

2002

Graduate College
Iowa State University

This is to certify that the master's thesis of

Denis Vitalievich Dorozhkin

has met the thesis requirements of Iowa State University

Signatures have been redacted for privacy

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my academic advisor, Dr. Judy Vance, for her continuous encouragement and guidance throughout my academic pursuits. The many opportunities she has given me during my stay at Iowa State University are greatly appreciated.

I would also like to sincerely thank Dr. Donald Flugrad and Dr. Thomas Rudolphi for being able to take on duties of my POS committee members.

Many thanks go out to all the VRAC students and personnel, especially those I got to work closely with, for the helpful advice and friendship they have given me.

Last, but by no means least, I wish to thank my parents, Vitaliy and Elena Dorozhkin, my sister, Masha Dorozhkina, and Dan and Marilyn Kammiller and their family. Without their unconditional love and support this thesis would not have been possible.

ABSTRACT

Spatial mechanisms offer a better alternative to electronically controlled multiple-input devices, such as robotic manipulators. Being purely mechanical devices, spatial mechanisms are less expensive, more reliable and more energy efficient. Furthermore, a single spatial mechanism is often capable of completing a motion task that would otherwise require several planar mechanisms to accomplish. Despite the potential benefits associated with operation of spatial mechanisms, development of such mechanical systems has been hindered by the lack of the appropriate mechanism design software applications.

This thesis presents the results of the research work that explored using virtual reality (VR) as a tool for designing spatial 4C mechanisms for rigid-body guidance. Spatial mechanisms operate within three-dimensional design space. The traditional human-computer interfaces (HCI) are limited to two dimensions and impose artificial constraints on the ability of the mechanism designers to correctly and efficiently specify the design problem and investigate the spatial mechanism synthesis results. Virtual reality provides a truly three-dimensional alternative to the traditional HCI's. The overall spatial layout of a mechanism design problem as well as the correct dimensions of the synthesized mechanisms are easily evaluated in the VR environments. This work combines such features as dynamic computation of fixed and moving congruences, application of branch and circuit defects filtering of the congruence planes, mechanism type identification at the solution selection stage, combined interaction with the guide map and the congruences, the option for either three- or four-position mechanism synthesis, and a speech control interface, to provide mechanism designers with an advanced VR spatial 4C mechanism design tool.

CHAPTER 1. INTRODUCTION

The term Virtual Reality (VR) refers to computer-generated three-dimensional (3-D) environments created by virtual environment (VE) systems, which can be interactively experienced and manipulated by the participants (Barfield and Furness, 1995). Stuart (2001) defines a VE system as a human-computer interface capable of providing "interactive immersive multisensory 3-D synthetic environments." In such systems the user's motions are tracked with position sensors and used to update the visual and auditory displays in real-time. This creates the illusion for the participants of being inside of the environment (Stuart, 2001). In addition to providing the ability to explore a design problem in three-dimensional space, VR environments often allow users to manipulate the objects in the environment in an intuitive way using a variety of instrumented gloves and wands.

The scientific and engineering communities have embraced virtual reality as a valuable tool because it offers a unique way to investigate data. Benefits of the VR systems are especially evident in the area of engineering product development, where these systems are used throughout the whole range of the product development cycle: from modeling and evaluation of the first prototypes, to providing training opportunities for end-product users (Bullinger et al., 1999, Deisinger et al., 2000, Oliveira et al., 2000).

This thesis presents the results of the research work that explored using VR as a tool for designing spatial 4C mechanisms. Spatial 4C mechanisms are two degree of freedom closed kinematic chains consisting of four links connected by cylindrical (C) joints. A cylindrical joint provides both translational and rotational movement along its axis. Unlike planar mechanisms that operate within two-dimensional design space, spatial mechanisms are

three-dimensional entities, which make the design and evaluation tasks especially well suited for the use of virtual reality technology. The overall spatial layout of a mechanism design problem as well as the correct dimensions of the mechanism are easily evaluated in the VR environments.

The benefits of using virtual reality applications for synthesis and analysis of mechanisms have been extensively explored at Iowa State University. Investigation of spherical mechanism design in VR has resulted in a series of programs, including SphereVR (Osborn and Vance, 1995), VEMECS (Kraal, 1996), and Isis (Furlong et al., 1999). Spherical mechanism design using VR has been expanded to include the design of spatial mechanisms. In 2001, the VRNETS program was created in order to aid in the synthesis of spatial 4C mechanisms (Kihonge et al., 2001). The research presented here builds upon the knowledge gained while using the VRNETS program.

The VRSpatial application was developed as a result of this research. It is a software package for designing spatial 4C mechanisms using virtual reality that offers the mechanism designers a wide variety of methods for defining the initial design problem, solving the problem, and evaluating the solution's feasibility. VRSpatial is written in the C++ programming language. Creation of the computer graphics objects and management of the VR environment is achieved with the SGI OpenGL™ Application Program Interface (Woo et al., 1999) and the VRJuggler virtual reality software library (Bierbaum et al., 2001). Computation routines from Florida Institute of Technology's SPADES software package (Larochelle, 1998) are used to perform mechanism synthesis and analysis. The user interaction method

consists of a position-tracked wand and a set of virtual menus, along with a speech recognition interface.

When compared to the functionality of the original VRNETS application, the new spatial mechanism design program offers its users an assortment of new and improved features. Some of the novel program characteristics are listed below:

- Implemented on a wide variety of VR systems, including Iowa State University's C6 facility.
- Improved methods of specifying the initial design problem, such as the ability to numerically adjust positions of the design objects, provide for more effective design.
- Multiple options are available for investigating the generated solution space, including dynamic congruence planes recalculation, the ability to select arbitrary lines from congruence planes, extension of the mechanism type identification and solution filtering options to the congruence planes, and the ability to investigate the guide map and the congruence planes simultaneously.
- Three-position mechanism synthesis functionality is provided.
- Speech command interface offers an alternative way of interacting with the application.

Chapter 2 of this thesis outlines some of the fundamental principles of motion synthesis of spatial 4C mechanisms. It also presents the history of computer aided mechanism design. Chapter 3 describes the program that can be used to design spatial 4C mechanisms in virtual reality, which was completed as part of this research. Chapter 4 explains the

program's functionality and its underlying mechanics in depth. Finally, Chapter 5 offers conclusions and recommendations for future work.

CHAPTER 2. COMPUTER AIDED KINEMATIC SYNTHESIS OF SPATIAL 4C MECHANISMS FOR RIGID-BODY GUIDANCE

The discipline of kinematics is concerned with investigation of geometric aspects of motions of a rigid body (or several connected rigid bodies) without consideration of the forces, causing the motions (Mallik et al., 1994). A rigid body is a set of points that always retain constant distance between any two of them. A mechanism can be defined as a collection of rigid bodies connected together with joints that constrain their relative motion. (McCarthy, 1990). The connections are designated as kinematic pairs, and every rigid body involved in the construction of a kinematic pair's is designated as a link. A sequence of links connected by kinematic pairs forms a kinematic chain, which can be either open or closed. In order for a kinematic chain to be classified as closed, every link in the chain must be connected to at least two other links, with one of the links in the chain being fixed. Furthermore, a simple kinematic chain is defined as a kinematic chain composed exclusively of binary links, that is, links that connect exactly two other links (Mallik et al., 1994). This thesis concentrates on the synthesis of simple single-loop closed kinematic chains, or mechanisms.

Mechanisms can be further classified according to the kinds of motion behavior they exhibit. A mechanism comprising links that move in planes parallel to the base plane and joints with axes that are strictly perpendicular to the base plane, is designated as a planar mechanism (McCarthy, 1990). If the joint axes intersect in a single point, thus forcing the links of the mechanism to rotate on concentric spheres about that point, the mechanism is called a spherical mechanism. Spherical mechanisms constitute special cases of spatial

mechanisms where the joint axes are skew to each other in space, allowing for general spatial motion.

In addition to the orientation relationship of the joints, motion of a given mechanism is related to the type or types of the kinematic pairs involved in its construction. There are six basic kinematic pairs: revolute (R), cylindrical (C), prismatic (P), spheric (S), screw or helix (H), and planar (P_L). If a single coordinate is sufficient for description of relative motion between a given pair of links, they are connected by a lower kinematic pair (Suh and Radcliffe, 1978). Such pairs have a single degree of freedom in contrast to higher pair connections that have more than one degree of freedom. The revolute pair, which is essentially a rotary hinge, and the prismatic pair, consisting of a linear slider, are examples of lower kinematic pairs. Normally, higher pairs can be reduced to the equivalent series of lower pairs (Suh and Radcliffe, 1978). A cylindrical joint (C) is an example of a higher pair obtained by mounting a hinge on a slider such that the axis of the hinge is parallel to the direction of the slider (McCarthy, 2000). This joint has two degrees of freedom and can be considered as a combination of a revolute and a prismatic pair.

In general, mechanisms are used to accomplish one of the following tasks: function generation, path generation, or rigid body guidance. The output motion of a mechanism designed for function generation is a function of the input motion. In the case of path generation, a point on a rigid body is guided by the mechanism through a sequence of points in space along a predetermined path. Finally, rigid body guidance, also known as motion generation, is responsible for guiding a rigid body through a series of finitely separated positions in space, designated as precision points (Suh and Radcliffe, 1978).

2.1 Spatial 4C Mechanisms

Spatial 4C mechanisms, investigated in this thesis, are simple closed kinematic chains consisting of four links connected by four cylindrical (C) kinematic pairs. They are formed when the floating links of 2 two-jointed open CC chains are rigidly connected to form a closed chain (McCarthy, 1995). In the case of motion generation, each open chain can be designed separately, provided that the floating links reach all of the prescribed design positions. The resulting spatial 4C mechanism has two degrees of freedom (McCarthy, 2000).

In the design of planar linkages, application of the Burmester theory for four-position synthesis results in the cubic curve of center points and the associated set of circle points. The center points serve as the set of location of the fixed joints of the RR chain, while circle points provide choices for location of the moving joints (McCarthy, 1994). In 1994 Murray and McCarthy demonstrated that Burmester's work could be generalized to include spatial CC chains (Murray and McCarthy, 1994). The spatial triangle technique is used to generate a set of parameterized lines that define the axes of the CC dyads that will guide the body through the four spatial positions. This set of lines is referred to as the fixed and moving congruences (Larochelle, 1995). A compatible pair of fixed and moving axes retains a constant normal distance and angle in each of the four design positions. In 1995 techniques for numerical determination of the congruences and for selection of lines from them were derived by Larochelle (Larochelle, 1995). His work provides the computational basis for this research.

Spatial 4C mechanisms synthesized using these techniques can be successfully assembled in each of four design positions. This method of selecting independent parameters from the fixed and moving congruences, however, does not eliminate mechanisms with circuit

or branch defects. Chase and Mirth define a circuit of a linkage as "all possible orientations of the links, which can be realized without disconnecting any of the joints" (Chase and Mirth, 1993). If more than one assembly is required in order to guide a mechanism through the specified design positions, the mechanism suffers from a circuit defect. A branch is a distinct configuration of the mechanism associated with a given position of the input link (Reinholtz et el, 1986). If more than one branch is associated with the prescribed design positions, the mechanism suffers from branch defect. In this case, it is possible that, while passing through a set of positions, the mechanism may experience a change in branch and enter a singular configuration. This will sometime cause the mechanism to fail because the input link is no longer capable of driving the output link (Tse, 2000). This research utilizes filtering algorithms that eliminate mechanisms with circuit and branch defects from the solution space.

2.2 Computer-Aided Mechanism Synthesis

Starting in the early 1950s, the discipline of mechanism design has undergone dramatic changes. Following the introduction and relative availability of digital computers in industry and university engineering programs, computer-aided design techniques have assumed an ever more important position among mechanism design tools (Erdman, 1985).

2.2.1 Planar mechanism synthesis and analysis programs.

Computer programs capable of mechanism analysis existed as early as 1951 (Kemler and Howe, 1951). However, the first application used to synthesize mechanisms was introduced in 1959 by Freudenstein and Sandor (1959). A punch card-driven IBM 650

computer was used to solve complex-number synthesis equations that resulted from reformulation of graphical-based techniques for computer solution in order to design path-generating four-bar linkages.

In the following two decades there has been an explosive growth in the number of computer-aided mechanism design programs. Such packages as IMP (Sheth and Uicker, 1972), ADAMS (Orlandea and Chace, 1977) and DADS (Wehage and Haug, 1982) were developed for the task of mechanism analysis. In the area of mechanism synthesis a special note should be given to KINSYN, developed at M.I.T in the early 1970s (Rubel and Kaufman, 1977). It is based upon computational principles outlined in the work of Freudenstein and Sandor (1959) and is the first program that used computer graphics to facilitate mechanism design. KINSYN used refreshable vector display devices to display the result of the synthesis process and allowed the users to graphically enter the design problem's parameters. Some of the other significant programs that were developed during that time period include LINCAGES (Erdman and Gustafson, 1977), which emphasized interactivity in the design process, and RECSYN (Chuang et al., 1981), which used solution rectification in order to eliminate solutions with circuit defects by graphically identifying the problematic solution sets.

The development of computer-aided mechanism design applications continues to be an important task in the area of kinematics and machine design. Networking has been implemented in the XLinkage program for Web-based analysis and simulation of planar mechanical systems (Katwyk and Cheng, 1997). Unlike most other mechanism design applications, XLinkage accomplishes the mechanism analysis task through a collection of small utilities, which are executed on multiple computers throughout the network. In

combination with mechanism design utilities, XLinkage emphasizes collaborative work among mechanism designers and distance learning students.

RealisMe, by Vasiliu and Yannou (2001), relies on artificial neural networks to synthesize dimensions of path-generating planar mechanisms as an alternative to the traditional graphical and analytical methods. In this approach, a very large number of mechanism cases for random dimensional values are generated through kinematic simulations, and the cases are learned by the neural network. Following the learning process, the network is capable of instantaneously producing an approximate solution to a given design problem, which then can be used in conventional dimensional optimization.

2.2.2 Spatial mechanism synthesis and analysis programs.

Spatial mechanisms can be used in manufacturing applications to provide a better alternative to electronically controlled multiple-input devices, such as robotic manipulators. Being purely mechanical devices, spatial mechanisms are less expensive, more reliable and more energy efficient (Myklebust et al., 1985). Nevertheless, until only a decade ago, the development of such mechanical systems was hindered by the lack of appropriate mechanism design software applications. The primary difficulty involves specifying the spatial mechanism design problem on the computer and in the correct interpretation of the resultant spatial solutions.

The earliest version of the spatial mechanism analysis program is the work by Coats and Cipra (Coats and Cipra, 1983), completed in 1983, in which they concentrated on animation of spatial linkages using IMP (Integrated Mechanisms Program). Computer-aided

mechanism analysis was extended to mechanism synthesis in 1985 with the development of the MECSYN-IMP-ANIMEC (Myklebust et al., 1985) software package. That work outlined some of the fundamental components required for a successful spatial mechanism design program, such as a high level of interactivity, effective and easy to use graphical input, and animation output with automatic 3-D geometric modeling that emphasizes rapid evaluation of the results. Characteristics of a successful graphics input interface for spatial mechanism design task were further refined in the Mechin application, developed in 1988 (Thatch and Myklebust, 1988). Mechin emphasized easy entry of the design specifications, such as positions, orientations and connectivities of joints, and positions and orientations of the bodies involved in mechanism synthesis. Input was achieved through an efficient layout of the screen, graphical 3-D positioning cues, and the ability to create and delete joints and links at any time, instead of having to specify that data sequentially around the mechanism loop.

The concepts outlines in the above-mentioned work became the foundations of the modern spatial mechanism design programs. In 1993, Sphinx, the first CAD program dedicated exclusively to the design of spherical four-bar mechanisms for rigid-body guidance was introduced (Larochelle et el, 1993). It provided such advanced features as three or four position mechanism synthesis, filters to eliminate most of the unusable mechanism configurations, and an intuitive user interface. To extend the range of potential users, it was later converted from an SGI workstation application into a Windows® program, resulting in SphinxPC (Ruth and McCarthy, 1997).

In 1998 Larochelle introduced Spades (Larochelle, 1998), an SGI-based program for design of spatial 4C mechanisms for spatial rigid-body guidance tasks. Either three or four

desired locations can be used to generate the mechanism solutions. Later version of Spades gained the ability to test the synthesized mechanisms for branch and circuit defects and to filter unacceptable linkage configurations.

The functionality contained in Sphinx, SphinxPC, and Spades applications was combined in the Osiris program, created in 2000 by Larochelle and Tse (Tse, 2000). Osiris is capable of generating both spherical and spatial mechanisms for two, three and four design positions. It offers advanced filtering techniques that have been extended to include three-position synthesis. Furthermore, it is portable to several computer platforms and allows users to graphically place desired locations in space using a spaceball input device.

2.2.3 Application of virtual reality to mechanism design.

The design of planar mechanisms is limited to two-dimensional space, so the traditional human-computer interfaces (HCI) of a monitor, a keyboard and a mouse are well suited for the task of the design problem parameter definition. However, operation of spatial mechanisms is associated with general 3-D space, and usage of a traditional HCI, even well designed, imposes artificial constraints on the ability of the mechanism designers to correctly and efficiently specify the design problem and investigate the spatial mechanism synthesis results.

Virtual reality provides a truly three-dimensional alternative to the traditional computer interface. Replacing the mouse and the monitor with a position tracked stereo visual display and a position tracked input device, VR allows the users to interact with the design problem by moving around and performing actions in 3-D space. The potential of

using VR technology in the design of spatial mechanisms was first recognized in 1995 by Vance and Osborn (Osborn and Vance, 1995), when the SphereVR program was created for analysis and synthesis of spherical 4R linkages. It required users to place coordinate frames on graphical representation of a sphere in the VR environment. The Newton-Raphson iterative approach was used to solve the non-linear equations, which resulted from Suh and Radcliff's displacement matrix mechanism synthesis method (Osborn and Vance, 1995).

Investigation of virtual reality as a medium for spherical mechanism synthesis continued in 1996 with the creation of VEMECS (Virtual Reality for MEChanism Synthesis) (Kraal, 1996). VEMECS relied on Sphinx algorithms for its mechanism analysis and synthesis functionality and essentially became a virtual reality interface to the Sphinx program. Following evaluation of the effectiveness of a VR interface compared to the traditional HCI methods (Evans et al. 1999), in 1999 Furlong et al. developed Isis, as the next generation spherical mechanism design tool (Furlong et al., 1999). Isis introduced the 'design in context' approach to the design problem definition, where digital models of the design part and of the work environment could be imported into the application and manipulated by the users instead of the conventional abstract coordinate frames. A real world design task was investigated and the resulting mechanism built by the designers.

In 2001 the spectrum of VR-based mechanism design applications was expanded to include analysis and synthesis of spatial 4C mechanisms, with the creation of the VRNETS program by Kihonge and Vance (Kihonge et al., 2001). Spades computation routines were used to provide the mechanism synthesis functionality of VRNETS. The program allowed users to investigate the design parameters associated with spatial 4C mechanisms, such as the

input design positions and the congruence planes, in a 3-D environment. Additionally, it provided the option of networking several instances of the application in order to facilitate a collaborative design process.

Operation and functionality of VRNETS has been explored by several mechanism designers. They discovered that while the program proved to be an effective tool in the synthesis and analysis of spatial 4C mechanisms, improvements and modifications to the program's structure and functionality were needed in order to take full advantage of the virtual reality design environment. The suggested changes were focused on improving the user interface, expanding the design problem specification functionality, providing higher degree of flexibility while working with the application, and improving solution evaluation methods. This thesis builds on the experience gained from operation of the VRNETS program, and incorporates the suggested changes into a new piece of software. New concepts and methods provide for a significant improvement over the previous spatial mechanism design program.

CHAPTER 3. OPERATIONAL DETAILS OF VRSPATIAL

VRSpatial has been developed at Iowa State University as a tool for the design of spatial 4C mechanisms in virtual reality environments. Computation routines from Florida Institute of Technology's SPADES software package (Larochelle, 1998) are implemented to perform mechanism synthesis and analysis. VRSpatial is written in the C++ programming language and uses the VRJuggler virtual reality software library (Bierbaum et al., 2001) and the SGI OpenGL™ Application Program Interface (Woo et al., 1999) for management of the VR environment and generation of computer graphics objects.

VRSpatial can be utilized on a variety of virtual reality hardware architectures, including projection-based displays such as the CAVE™ and the MD Flex (Mechdyne, 2002) and helmet-mounted displays, such as the V8 HMD (Virtual Research Systems, 2002). A wide array of tracking systems and input devices can also be used. The primary operating environment for VRSpatial throughout its development process has been Iowa State University's C6 system. The C6 is a 10'x10'x10' virtual reality room composed of six rear stereo projective screens that surround the user to provide a fully immersive experience (Figure 3.1). The C6 is powered by a Silicon Graphics® Onyx2® computer with six Infinite Reality2™ graphics display generators, 24 R12000 processors, and 12 gigabytes of memory. VRSpatial is designed to take full advantage of the C6 hardware configuration, although it can also be used with other less powerful VR environments.

This chapter describes the VRNETS program, which preceded this research work, the reasons for developing the VRSpatial application, the development process itself, and

VRSpatial's structure and functionality. The procedure for designing spatial 4C mechanisms in a virtual reality environment is also outlined.

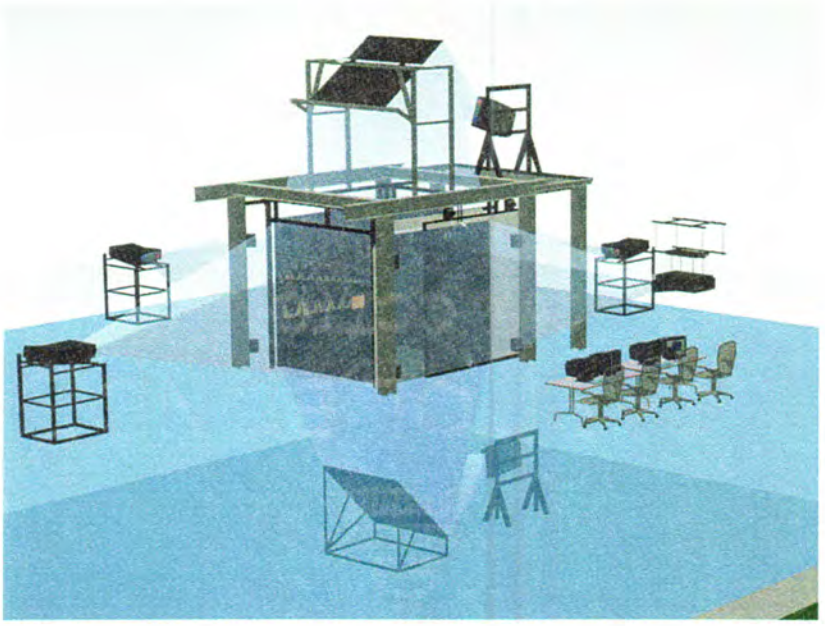


Figure 3.1: Iowa State University's C6 Facility.

3.1 VRSpatial Development Background

Kihonge and Vance (Kihonge et al., 2001) developed VRNETS as the first application for design and analysis of spatial 4C mechanisms in virtual reality. VRNETS relied on Engineering Animation WorldToolKit libraries (Engineering Animation, 1998) for simulation management and WorldToolKit Immersive Display Option (WTKIDO) libraries for management of the virtual reality environment. Mechanism synthesis and analysis routines from the SPADES program were used for computation tasks. VRNETS was designed to be displayed in the Iowa State University's C2 facility - a CAVE-like virtual reality room with three rear projection screens for each of the three walls and a direct projection screen for the floor. The C2 provided users with the ability to investigate the VR environment directly in

front of them, to the right, to the left and below. Ascension Technology Corporation's Flock of Birds® magnetic tracking system (Ascension, 1993) was used to track the position and orientation of the user's head and the input device in the environment. The interaction was performed using a Fakespace PINCH™ Glove (Fakespace, 1995) that registered contact between user's fingers to provide command input to the VRNETS application.

The functionality of VRNETS allowed the users to walk into the VR environment, define the mechanism design problem by placing four instances of an object in the environment using natural three-dimensional hand movements, generate a solution, and analyze it. The C2 environment could accommodate multiple users at once, thus allowing several mechanism designers to participate in the design process (Figure 3.2). In addition, some networking functionality was implemented in VRNETS, providing the users at remote sites with the ability to collaborate in the design of spatial 4C mechanisms.

Figure 3.2: VRNETS in the C2 Facility.

The VRNETS application has proven to be an effective tool in the process of designing spatial 4C mechanisms. Nevertheless, during the operation and evaluation of the program, users have offered several suggestions for its improvement. These suggestions included changing and expanding the interaction functionality and making the mechanism design process more flexible and efficient. One of the most obvious drawbacks of the application was the inability to investigate the design space in its entirety, since the C2 could not provide the users with a view of the space above and behind them. This limitation is most clearly apparent when users view and select sets of congruence planes. These planes are spatial in position and orientation and can be distributed throughout the whole design space. Even though the users could reposition the VE in order to view previously hidden components in the environment, they expressed the desire to be fully surrounded with the design space.

VRSpatial was designed for display in fully immersive VR systems. Moving the application to the C6 provided a fully enclosed virtual environment, surrounding the user with stereo computer images during the spatial mechanism design process. The move to the C6 necessitated a change in software from the World Tool Kit-based VRNETS to VRJuggler. VRJuggler has full support for the C6 system. In addition, VRJuggler provides the ability to utilize a wide variety of VR systems, from desktop displays to HMDs to CAVE™s. In order to take full advantage of VRJuggler's capabilities and to accommodate changes suggested by the users, a decision was made to create a new version of the software for design and analysis of spatial 4C mechanism in virtual reality. The new application would retain the core features of VRNETS but include additional functionality, which would provide the designers with

more information throughout the design process. Additionally, new and improved mechanism computation routines from Florida Institute of Technology were to be implemented.

3.2 Interaction Methods in VRSpatial

The primary interaction device in the C6 system is the wireless wand. The wand has two push buttons and a pad that provides four additional push selectors (Figure 3.3). A six degree of freedom position sensor is attached to the wand. Another position sensor is attached to a pair of NuVision shutter glasses. The position and orientation information from the two sensors are relayed to the application via a wireless data transmitter. This information is then used to compute the images displayed on the walls of the C6 with correct perspective for the user with the tracked glasses and to determine if there is interaction between the wand and the virtual objects. In order to provide a visual representation of the wand in the VR environment, a semitransparent cube is drawn to appear at the tip of the wand. Objects that can be manipulated with the wand will be affected only if they are located within the selection cube. To access regions of the virtual environment located beyond the physical space of the C6, a simple navigation method has been implemented. By depressing the bottom section of the mouse pad (button 5) on the wireless wand, users can translate in the direction that the wand is pointing.

Figure 3.3: Wireless RF Wand.

VRSpatial provides a wide variety of functionality options to the users. To allow for an effective way to access that functionality, a system of virtual menus has been developed. A dedicated menu object class and a set of associated functions provide the users with menus that can be dynamically created, reconfigured, and deleted during the program's execution. This allows for a higher degree of flexibility compared to the traditional VR menu systems that rely on static texture-mapped objects to represent the menus and, thus, cannot be easily changed while the program is running.

The main menu system can be displayed at any time during the program's execution by depressing button 2 on the wireless wand. The location of a menu is determined from the position and orientation of the wand, so that the menu appears attached to the wand. This allows the users to keep the menu system from being obstructed by the objects in the environment. Once a menu is displayed, the primary function of the wand buttons is to allow

navigation within the menu system (presented in Appendix A). Wand buttons 4 and 6 are used to toggle between the available menus, and buttons 3 and 5 are used to scroll up and down within the current menu. The current selection is identified with a marker drawn to the left of the selected option in the menu (Figure 3.4). Once the desired option is selected, button 1 or the wand's trigger will activate that menu option.

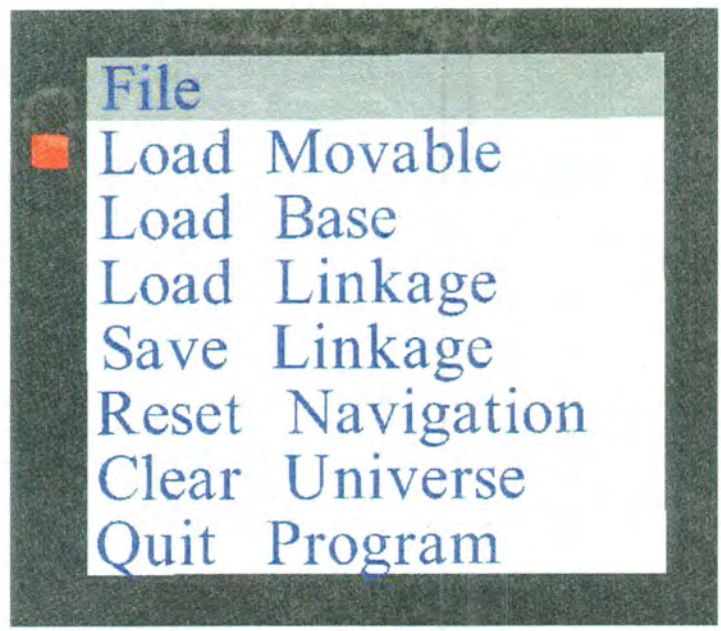


Figure 3.4: VRSpatial Main Menu.

3.3 Definition of the Design Problem

Four precise positions must be specified in order to define a spatial 4C mechanism design task. Continuing the 'design in context' approach of Isis and VRNETS, VRSpatial allows the users to load and manipulate models of an object in the VR environment instead of working with abstract coordinate systems. The desired object is selected in the 'Load Movable' file menu. Once the file is loaded, four copies of the object are introduced into the environment by successively pressing button 3 on the wand. When the button is depressed, a

semitransparent copy of the object model appears attached to the wand. Figure 3.5 shows a user in the virtual environment introducing the design object into each of the four positions.

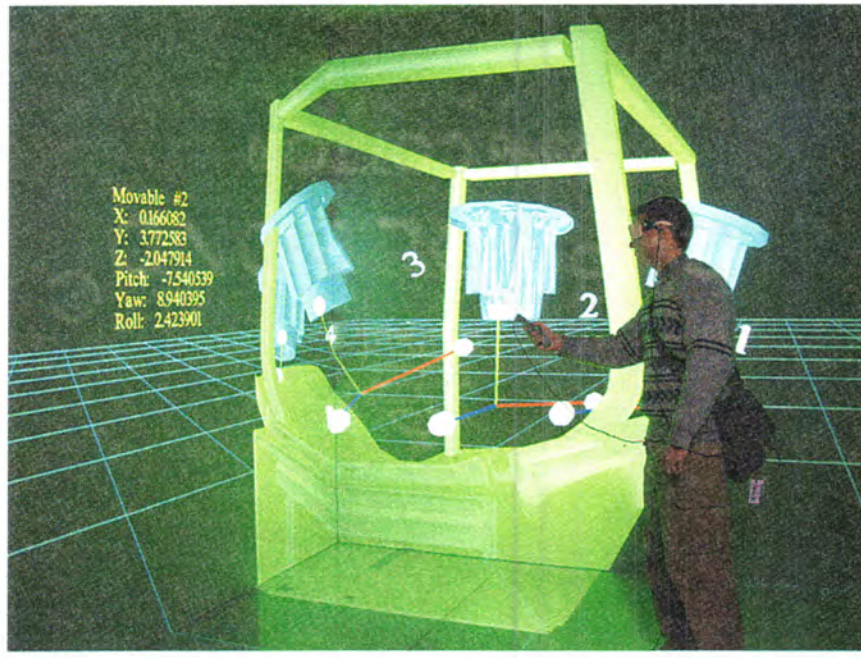


Figure 3.5: Object Manipulation in VRSpatial.

Each object can then be placed in a desired location by moving the wand and/or navigating and then releasing the button. The object turns opaque when placed in its static condition. Each object is numbered according to the order in which it was introduced into the environment. The objects can be repeatedly selected and repositioned by placing the wand selection cube over the origin of the object's coordinate system and depressing the trigger button. During the selection phase, the position of the object with respect to the wand is preserved while the part is repositioned instead of 'snapping' the model to the coordinate system of the wand. These adjustments can take place before the next object in the design sequence is introduced into the environment. Additionally, the orientation of the object

around a single coordinate axis can be adjusted by grabbing the sphere located at the end of that coordinate axis and rotating the wand around the axis.

In order to counter possible imperfections in the tracking system as well as the inherent inaccuracy of the user's hand motions, the location of each object can be also adjusted numerically by repeatedly incrementing or decrementing one of the position parameters (i.e. x, y, or z-coordinate) by a small value. This option is available once all of the four design positions have been defined. The option is utilized by depressing button 3 while selecting an object. The user is then presented with the position adjustment menu where he or she can select the desired parameter, increase or decrease its value by pressing the right and left mouse pad sections respectively, and then either accept or discard the changes (Figure 3.6).

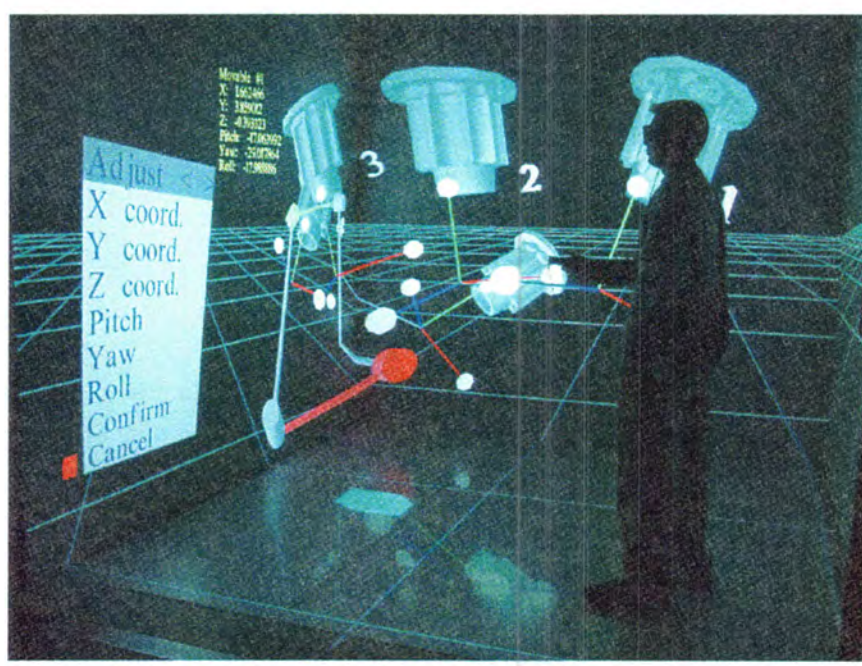


Figure 3.6: Numerical Adjustment of the Design Positions.

The user is made aware of the precise numerical values describing the position and orientation of each object through a heads-up display. Whenever an object is selected, information about the object's number in the design sequence and its XYZ coordinates along with rotations around the XYZ axis are displayed in front of the user's eyes. Positional data from the user's tracked glasses is used to place the text objects, containing the information, several feet away from the glasses. The information is dynamically updated as the object is being manipulated (Figure 3.5).

In order to provide for maximum flexibility while defining the design problem, the position sequence can be reset by the user and redefined by selecting the objects in the desired succession. In addition, an instance of the object can be deleted at any time and subsequently reintroduced into the environment. The sequence numbers are adjusted to account for the changes. The representation of the objects' coordinate systems can be hidden once satisfactory locations have been defined, eliminating unnecessary geometry from the design environment and allowing for better visual analysis of the synthesized mechanism.

Since a real-life mechanism is likely to operate in a physical environment where constraints will most likely limit its range of motion or affect the precise path that the moving part should take, a model of the working environment can be loaded as well. In this case, the models of static objects in the environment are loaded and initially appear at the origin of the world coordinate system. Once they are introduced into the virtual design space, these models can be manipulated in the same manner as the input design positions if the default location is not satisfactory.

3.4 Generation and Selection of the Solutions to the Design Problem

Once the four design positions have been specified, users have the option of generating the solution to the given motion generation task. The set of all possible solutions to the mechanism synthesis problem can be represented by either the fixed and moving congruences (Larochelle, 1995) or the guide map.

3.4.1 Congruence planes.

The congruences are infinite sets of lines that lie in planes and define the fixed and moving axes of the joints of a spatial 4C mechanism capable of guiding a body through the four defined design positions in space. They are represented in the virtual environment by finite-sized red and yellow planes that correspond to the fixed and moving congruences respectively (Figure 3.7).

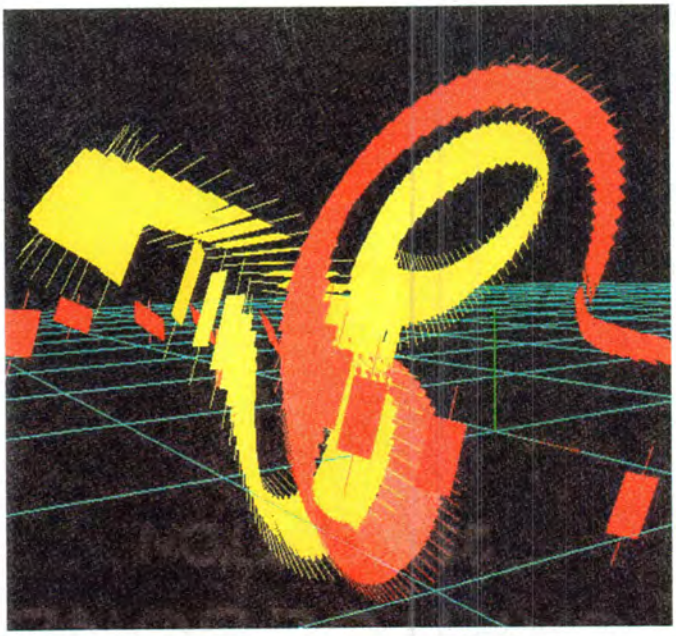


Figure 3.7: Fixed and Moving Congruences in VRSpatial.

The moving congruences define joint locations of the mechanism assembled in the first design position. Since the moving axes of the joints occupy various positions as the mechanism moves, the default location for display of the moving congruences is their location as the mechanism is configured in the first design position. The design positions define the location of the congruence planes.

A generator line is associated with each congruence plane and has the same color designation as the plane. Every line that lies within the given congruence plane and is parallel to the generator line associated with that plane is a valid congruence line. Congruence lines are associated in pairs. One fixed congruence corresponds to only one moving congruence and one pair of fixed and moving congruence lines defines a CC dyad of a mechanism with axes that will maintain constant normal distance and relative angle in all of the four prescribed design positions. Two CC dyads fully define a mechanism. Therefore, the user must select two congruences to define the mechanism.

Once the set of congruence planes is displayed in the environment, the user defines either the driving or the driven dyad of the mechanism by choosing the appropriate option from a menu. Selection of a congruence line is performed by intersecting the selection cube with a generator line in either a fixed or a moving congruence plane and pressing the trigger button. The generator line in the given plane and the compatible line from the other set of congruence planes change their colors to either green or blue. If the driving dyad is being designed, then the lines are colored green. If the driven dyad is being defined, then the lines are colored blue. The second dyad of the mechanism is defined in the similar fashion. The selection procedure can be repeated until a good solution is obtained. At that point the

designed mechanism can then be displayed in the design environment by selecting the "Show Linkage" option from the menu.

Since there is an infinite number of valid congruence lines, an arbitrary line within a congruence plane, other than the default generator line, can be selected. This is achieved by depressing button 3 on the wand while intersecting the selection cube with a generator line and then moving the wand in space, changing the location of the congruence line within the plane. The button is released at the desired location and the corresponding congruence line is displayed.

If the available fixed and moving congruences are found unsatisfactory, there might arise the need to create a new set of solutions by modifying one or more of the initial design positions. As the design positions of the object are moved, the congruence planes are dynamically recalculated and redisplayed.

3.4.2 Guide map.

The guide map is an alternative representation of the solution space. It is a two-dimensional color-coded plot, which allows mechanism designers to generate a mechanism with a single selection from the plot, while providing information about the type of mechanism that is being synthesized. In order to generate a guide map, the problem is solved and a set of the fixed and moving congruence lines is generated as usual but not displayed in the environment. As mentioned earlier, selecting two congruence lines selects the two CC dyads of the spatial 4C mechanism. These two independent selections can be mapped to the X- and Y-axes of the guide map. In addition, each possible solution is classified according to the

mechanism type of its spherical image - a spherical four-bar mechanism with link lengths equal to the angular twist of the links of the spatial 4C mechanism that is being considered (McCarthy, 2000). A color chart is displayed alongside the guide map to help designers in the mechanism identification process (Figure 3.8).

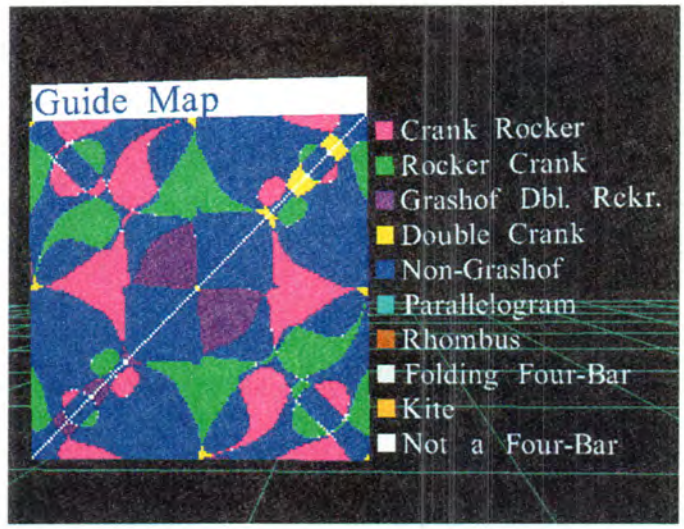


Figure 3.8: Guide Map.

When it is initially introduced into the design environment, the guide map is positioned several feet away from the user's eyes. Its location and orientation can then be altered by depressing button 3 on the wand and moving the wand in space. In order to select a point on the guide map the user presses the trigger button. A laser-like pointer is drawn from the tip of the wand and the selection is performed by intersecting the pointer with the guide map at a desired location and releasing the trigger. The selected point is identified with two intersecting lines and the corresponding mechanism is automatically synthesized and displayed in the environment. Selection from the guide map can be performed repeatedly until a satisfactory mechanism is obtained.

Furthermore, the solutions presented on the guide map can be filtered in order to eliminate mechanisms with branch and circuit defects. The circuit defect is present when a mechanism has to be disassembled and then reassembled in order to reach all of the design locations (Tse, 2000). A mechanism suffers from a branch defect if it enters a singular configuration where the input link is not capable of driving the mechanism (Tse, 2000). Filters can be applied to the guide map by selecting the appropriate menu option. Points on the guide map that correspond to the mechanisms with these defects are shown dimmed on the guide map. In Figure 3.9, most of the guide mapped solutions would exhibit branch or circuit defects as indicated by the large dimmed areas on the map.

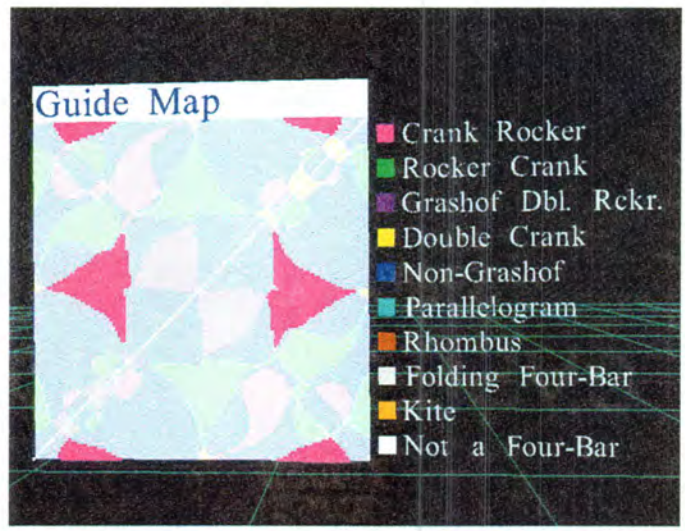


Figure 3.9: Guide Map with Applied Filters.

3.4.3 Simultaneous interaction with congruences and guide map.

While the guide map contains valuable data regarding the type of mechanism that is being synthesized and the possible defects, it lacks the three-dimensional information associated with congruence planes drawn in the design environment. Therefore, an option is

provided to display and interact with the congruences and the guide map simultaneously. Selection from the congruence planes is performed in the usual fashion. When a driving dyad is selected from the congruence planes, this corresponds to selecting a value along the x-axis of the guide map, and a red vertical line is drawn on the guide map. A horizontal line is drawn if a driven dyad is selected from the congruence planes (Figure 3.10).

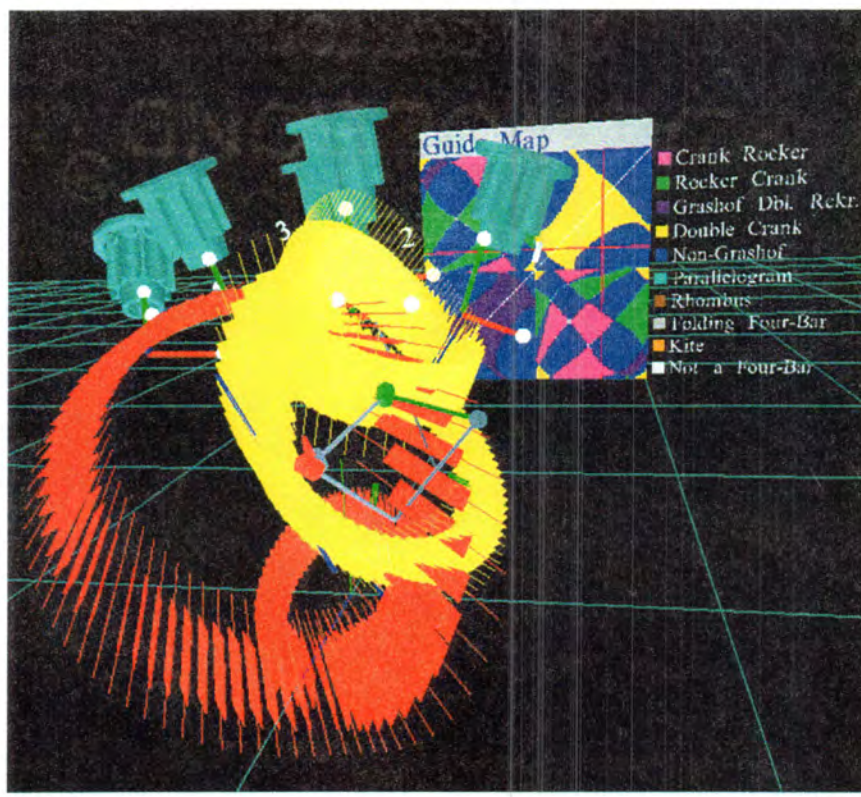


Figure 3.10: Interaction with Congruences and the Guide Map.

The intersection point of the two lines on the guide map identifies a valid mechanism solution. This solution is identical to the selection from the congruence planes. Once desired dyads are identified, the mechanism can be displayed in the environment with the "Show Linkage" menu option.

If the wand trigger is pressed while either in the driving or the driven dyad selection mode, but no generator line is detected within the vicinity of the wand, the application switches into the guide map selection mode - a laser pointer is drawn and a point on the guide map is selected as usual. In this case, the corresponding lines within the congruence planes are determined and colored according to driven or driving dyad selection, and the mechanism is synthesized and displayed in the environment.

Furthermore, the mechanism identification and filtering functionality of the guide map has been extended to the fixed and moving congruences. If this option is enabled, selection of the first dyad of the mechanism from one of the congruence planes will cause the rest of the planes to become color-coded. The color of each plane corresponds to the type of the mechanism that will be synthesized if the second dyad is selected from the given plane (Figure 3.11).

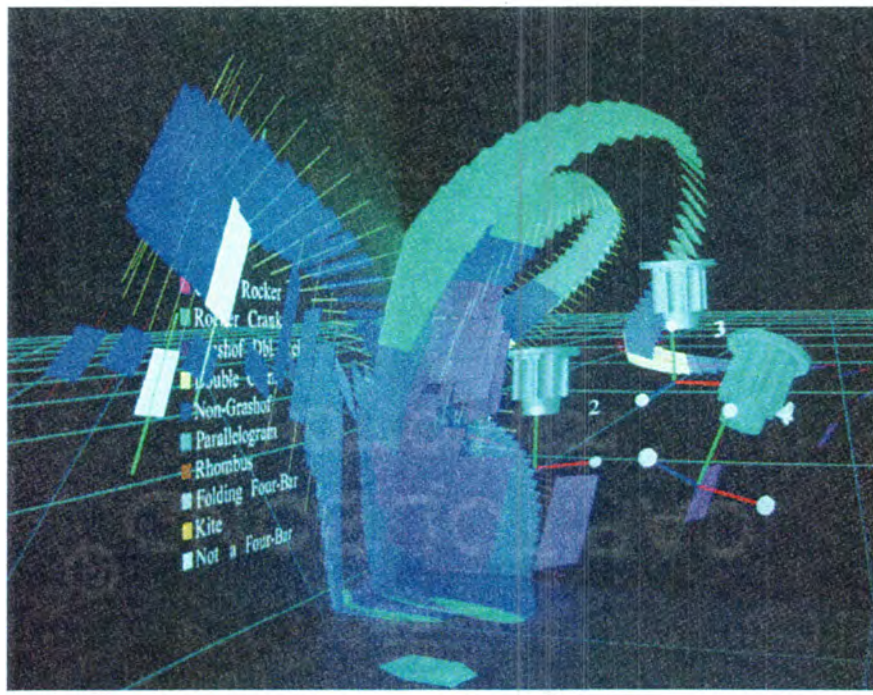


Figure 3.11: Color-Coded Congruence Planes.

With filtering enabled, the congruence planes will also be either semitransparent or opaque, corresponding to mechanisms with and without defects respectively. This capability to interact with the guide map and congruence planes simultaneously provides the designer with increased flexibility in choosing a final solution.

3.5 Three Position Mechanism Synthesis

Four fully-defined design positions are required in order to generate the sets of fixed and moving congruence planes and the associated guide map. The generated solution space provides many possible mechanism solutions to guide the work part through the four given positions. However, when a spatial mechanism is being designed to accomplish a real-world task, such as transporting a part from one machining station to another station in the environment of a manufacturing plant, quite often only two out of four positions are critical during the transportation phase. These positions would represent the initial location and orientation of the work part at the first machining station and the final location at the station where the part is being placed. The other two design positions are arbitrarily placed by a mechanism designer in order to completely define the design problem. While the two additional positions provide some information about the desired path that the work part will follow once the mechanism is animated, it is feasible to assume that there are situations where having four positions simultaneously displayed in the design environment would have little or no benefit to the designer. In order to account for such situations and to provide mechanism designers with another tool in mechanism synthesis process, VRSpatial has an option to synthesize a valid spatial 4C mechanism using only three of the initial design positions.

In order to generate a spatial CC dyad compatible with three given design positions, either the fixed or the moving axis of the dyad has to be defined in space by the user. These user-defined axes are called design axes. The corresponding second axis (moving or fixed, respectively) of the dyad is then computed. Since two dyads are necessary in order to fully define a spatial 4C mechanism, the designer has to locate two design axes in the virtual environment. Once animated, the resulting mechanism will guide the part through positions 1, 2, and 3.

The process is initiated by selecting the Three-Position Synthesis option from the Position Synthesis menu. No valid parameters for the two design axes have been specified by the user at this point. To begin the design process, the fixed driving axis is placed through the point (0.0, 0.0, 0.0), with a direction vector of {-1.0, 0.0, 1.0}. This axis is represented in the virtual environment as a solid wide line of green color. Similarly, the driven dyad's fixed axis is placed through the point (0.0, 1.0, 0.0), with a direction vector of {1.0, 0.0, 1.0}. It is drawn in a solid blue color. The spatial 4C linkage that results from these axes choices is also displayed in the environment. Simultaneously, the user is presented with a menu that can be used to adjust the location and direction of the two axes if necessary, and a heads-up display, where these parameters are shown. If the user chooses to design using the moving axes or a combination of moving and fixed axes, then the menu allows the user to change the design axes to the desired type. If the moving axes are the design axes, they are represented by a stippled pattern. As these changes are being made, the linkage is re-computed and re-displayed in the environment.

Instead of using the menu a manual line adjustment mode is provided. Each design axis is manipulated by intersecting the selection cube with the axis' graphical representation and depressing the trigger on the wand. By moving the wand, the desired location and orientation of the design axis in space are then set by the user. As in the case of the menu-assisted adjustment process, the numerical information is made available through the heads-up display. If a moving axis of a dyad is desired instead of the fixed one and vice versa, the axis type can be toggled by intersecting the selection cube with the axis and pressing button 3 on the wand. The graphical representation of the axis is updated accordingly (solid vs. stippled). Again, the linkage is re-computed in real time as these adjustments are made. In order to terminate the manual adjustment mode, the main menu button is pressed (button 2). The three-position synthesis menu is displayed, and the user can either continue adjustments using the menu or confirm the new parameters and quit.

3.6 Mechanism Evaluation

Once a spatial 4C mechanism is synthesized and displayed in the design environment, designers can investigate the mechanism by moving around in the virtual reality system or using the navigation option of VRSpatial. The mechanism is drawn using a set of cylinders that represent the axes and the links of the mechanism. In order to draw part of the mechanism geometry in the VR environment, a cylinder of unit height and diameter is translated, rotated and scaled according to the computed location and orientation of a given axis or link. Different segments of the mechanism can be color-coded according to their functionality by selecting the Color Linkage option from the Preferences menu. The driving link is drawn in

green and the driven link is drawn in red. The fixed link, the coupler, and the coupler attachment are drawn in the default gray color.

To verify that the synthesized mechanism is capable of completing the task for which it was designed, that is, moving the object through the three or four design positions, the mechanism can be animated by the user. The animation is performed by incrementally varying the two input parameters of the driving link: the input angle and the input displacement. The positions of the mechanism links are calculated for each set of input parameters and then used to draw the mechanism in the VR environment.

The animation process is initiated by selecting the appropriate option from the Animation menu. Once the mechanism is animated, users can observe its operation and check for collisions between parts of the mechanism and between the mechanism and the environmental objects. During animation, the objects displayed at the input design positions are drawn semi-transparent, while the object attached to the coupler is drawn opaque. The animation speed can be changed by selecting the 'Adjust Speed' option and wand buttons 4 and 6 can be used to accelerate or decelerate the animation respectively. Change in the animation speed is achieved by varying the incremental values for the driving link input parameters. The animation can be also reversed. Furthermore, users have the ability to constrain one of the input parameters and investigate the behavior of the linkage when only the input translation or only the input rotation are being varied.

Once an acceptable mechanism is synthesized, it can be saved to an output data file. This is done by selecting the 'Save Linkage' option in the File menu. A dedicated folder within the VRSpatial installation directory is used to store the mechanism files. The files are

numbered sequentially as they are being saved by the program. During a save, the folder is scanned for the presence of the existing files and the next available file number is assigned to the current mechanism file, i.e. "linkage7.vrnets". The user is then presented with a message that confirms the save and provides the number of the file. An existing file can be opened by choosing the 'Load Linkage' option and then selecting one of the available file names. The data file contains all of the relevant information associated with a linkage, including the axis and links parameters, mechanism type, design positional parameters, and animation data. When a data file is loaded into the design environment, the state of the design session is restored to its precise state at the time of the save, except for the models of the work object located at the design positions. The design positions at this time are represented by simple coordinate systems, and the object models can be re-introduced into the environment by loading one of the available geometry files. If a linkage file could not be successfully loaded by the application, for example, because of the invalid file contents or insufficient file access privileges, the appropriate message is generated for the users.

If the user desires to investigate a new design problem, the Clear Universe option can be chosen from the File menu. All of the VRSpatial state variables are reset to the default values, and the environment is cleared of the existing objects and prepared for the next design session.

3.7 Speech Control

The VRSpatial application has significantly more tools and design options when compared to the original VRNETS program. Most of the program's functionality is heavily

dependent upon input from the users. The main interaction method is via the menu system, described previously. While VRSpatial menus provide complete control over the application's operation, often the user is required to make several consecutive selections from interlinked menus to access a certain function. One such case is animation speed adjustment, where three menu choices are required in sequence to accelerate movement of the mechanism: "Animation"→"Adjust Speed"→"Go Faster". Each selection requires a certain amount of time to move through the menu system and select the desired option. Changing the hierarchy of the menu system would reduce the number of consecutive selections, but it would also lead to overly cluttered menus, increase in navigation time, and, ultimately, reduction in the interaction efficiency. A speech control interface has been implemented instead to provide an alternative method to control the VRSpatial application.

A dedicated Windows® computer is responsible for providing the speech recognition capabilities. It runs a speech interface that has been created with IBM ViaVoice™ for Windows® Release 8 Professional Edition, IBM alphaWorks Speech for Java and the omniORB2 version 2.8 of the CORBA standard. Audio input to the computer is provided via a wireless microphone worn by the user while interacting with the program. The microphone's stationary receiver is located in the C6 facility, so the audio signal is routed to the speech recognition station via a system of communication hardware and software components, called Matrix, that provides the ability to direct video and audio signals between components of the system. This system is also used to feed the audio output from the speech interface into the C6 speaker system. A local area network (LAN) is used to communicate the

recognized speech commands to the SGI machines responsible for operation of the main VRSpatial application.

If the speech recognition functionality is desired during a mechanism design session, the VRSpatial application is executed normally. Once the application is running, the user starts the speech recognition client on the Windows® computer. In order to provide the user with the ability to remotely control the operation of the Windows® machine from the C6 facility, Virtual Network Computing (VNC) from AT&T Laboratories has been installed on the machine. Once the speech recognition client has successfully established a data connection with the VRSpatial application, an appropriate message is generated on the SGI machine. At this point, the system is ready for voice commands. The voice control can be toggled at any time by selecting the "Voice Control" option from the "Preferences" menu in VRSpatial.

During operation of the speech interface, the numerical value of the current level of the input audio signal is continuously displayed on the Windows® machine and used to confirm correct operation of the system. If the user issues a valid VRSpatial command, the speech interface generates a message, identifying the command, and sends a set of arguments to the appropriate function within VRSpatial. The set of arguments is analyzed and the appropriate actions are undertaken in order to fulfill the user's request. In some cases these actions are the same that take place when a menu selection is made, while in other cases, a different combination of actions has to be executed. The latter situation might occur when a command is being executed that would otherwise be available only after several consecutive selections from the menu system with certain actions taking place after each selection.

The main purpose of the speech interface implementation in the VRSpatial application is simplification of the interaction method. The ability to control the application using the existing set of menus is preserved, even though most of the menu system functionality could be effectively supplied with the speech interface. Speech control provides the application users with the option to access almost any point in the menu selection sequence with a single sentence, greatly improving the interaction efficiency. Details of the implementation of the speech control will be presented in the next chapter.

CHAPTER 4. IMPLEMENTATION DETAILS OF VRSPATIAL

VRSpatial is written in the C++ programming language. OpenGL™ API (Woo et al., 1999) and the VRJuggler virtual reality software library (Bierbaum et al., 2001) are used to generate the computer graphics objects and to manage the VR environment. Synthesis and analysis of mechanisms is based on computation routines from SPADES software package (Larochelle, 1998). This chapter outlines the implementation details of the VRSpatial application.

4.1 Menu System

A stand-alone menu object class and functions have been developed. This menu class has the ability to initialize a new instance of the menu object or change contents of an existing menu object at any point in the program's execution, except when the menu is being displayed in the VR environment. The ability to change the menu object's content at runtime is used in VRSpatial to update the information displayed in the file manipulation menu to reflect the current state of the system's directories.

During the initialization phase a title is assigned to the menu. The menu contents are built by specifying the string designator associated with each option ("Confirm", "Load", etc.). The options are numbered sequentially as they are being added to the menu object. The menu object is automatically resized in order to accommodate the title and option names of different lengths and different number of options. The menu geometry in the VR environment is created using OpenGL primitives and the GLF library (Podobedov, 2002). GLF allows for

display of two- and three-dimensional text in OpenGL, with a variety of supported fonts and display options. In order to attach the menus to the wand in the VR environment, the 4x4 transformation matrix associated with the position and orientation of the wand is post-multiplied by a translation transformation matrix, which moves the menu several feet away from the wand. GLF-based text is also used to generate the heads-up display that provides positional information of the objects to users.

4.2 Object Manipulation

Such graphical entities in the VRSpatial environment as the design objects and the environmental objects are generated using geometry files saved in the Videoscape 3D Object File Format (.geo). A dedicated VRJuggler file loader loads the geometry files. Once loaded into the application, a (.geo) file becomes one of the attributes of the 'object' class. In addition to the geometry data, the 'object' class contains such information as the X, Y and Z coordinates of the object in the VR environment, its rotation angles around each of the coordinate axes, its design sequence number, its status – selected or not, etc. Every effort was made to make the manipulation of the objects to be as close to real-world interaction as possible. At the time the object is introduced into the environment, its 4x4 transformation matrix is set to the values provided by the wand's transformation matrix, causing the object to appear 'attached' to the wand. Once released, the object can be repeatedly 'grabbed' or selected by the user. During each selection and subsequent repositioning of the object, its current location and orientation with respect to the wand is preserved. The following formula

is used at the selection moment to determine the relative 4x4 transformation matrix, $[M_{rel}]$, which represents the location of the object in wand's coordinate system:

$$[M_{rel}] = [M_{w\_sel}]^{-1}[M_{obj}], \tag{4.1}$$

where $[M_{w\_sel}]$ is a 4x4 transformation matrix representing the location of the wand in the C6 coordinate system at the selection time (provided by the position tracker) and $[M_{obj}]$ is a 4x4 transformation matrix representing the location of the object in world coordinates at the selection time (stored in the 'object' class).

The inverse of the $[M_{w\_sel}]$ matrix is calculated using VRJuggler matrix manipulation functions. Once the $[M_{rel}]$ matrix is determined, the $[M_{obj}]$ matrix is continuously updated while the object is being adjusted using the following expression:

$$[M_{obj}] = [M_{wand}]\ [M_{rel}] \tag{4.2}$$

where $[M_{wand}]$ is a 4x4 transformation matrix representing the current location of the wand in the C6 coordinate system (continuously provided by the position tracker).

A similar formula is used to calculate the location of the wand in the object's coordinate system. This is used when the user rotates the object around a single axis, to ensure that the physical motion of the wand with respect to the object realistically corresponds to the changes in angles of the object's rotations around its axes.

Navigation in VRSpatial is provided by applying a global transformation matrix $[M_{nav}]$ before any of the world's objects are drawn. As long as the navigation button is depressed (button 5), the translational component of the $[M_{nav}]$ matrix is changed by a small fixed value every time a frame is rendered (Equation 4.3).

$$[M_{nav}] = [M_{shift}]\ [M_{nav}] \tag{4.3}$$

where $[M_{shift}]$ is a 4x4 matrix containing translation transformation in the direction opposite to the wand's direction. This creates the illusion of 'flying' forward. At the program's startup, or after a navigation reset, the coordinate systems of the virtual world and the C6 are collinear, but once the user has navigated from the starting point, direct utilization of the wand's and objects' transformation matrices can no longer be used in the selection process. In this situation, additional calculations are performed in order to account for the shift in coordinate systems:

$$[M_{wand}]' = [M_{nav}]^{-1}[M_{wand}] \tag{4.4}$$

where $[M_{wand}]'$ is a 4x4 transformation matrix representing the location of the wand in the shifted world coordinate system, $[M_{nav}]$ is the navigation matrix, and $[M_{wand}]$ is a 4x4 transformation matrix representing the location of the wand in the C6 coordinate system (provided by the position tracker).

4.3 Congruence Generation and Interaction

The result of the congruence computation process is a set of directed screw lines combined in groups of four pairs of parallel lines. Figure 4.1 depicts a directed screw line, which is, essentially, a line in space, defined by two vectors: the unit directional vector $\vec{S}$ and the moment vector of the line with respect to the origin of the local coordinate system $\vec{S}_0$ (Tse, 2000). The moment is defined as $\vec{S}_0 = \vec{c} \times \vec{S}$, where $\vec{c}$ is a vector from the origin to any point on the directed screw line. These two parameters are also known as the Plücker coordinates of the line (McCarthy, 2000) and can be written in the following way:

$$S = \begin{Bmatrix} \vec{S} \\ \vec{c} \times \vec{S} \end{Bmatrix} \tag{4.5}$$

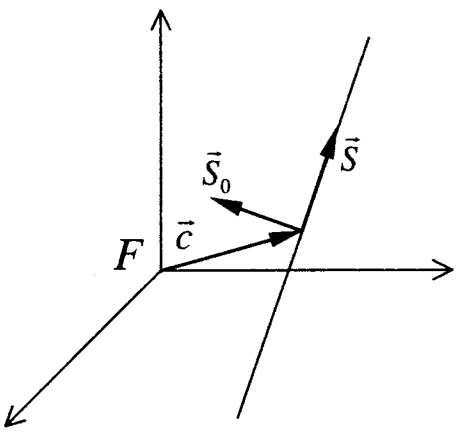


Figure 4.1: Directed Screw Line **S**.

Every pair of parallel directed screw lines uniquely defines a plane in space. Consequently, every eight directed screw lines will define four planes in space, two of which are the fixed congruence planes and two are the moving congruence planes. The number of these groups of screw lines, and, therefore, the total number of congruence planes displayed in the environment, is determined by the solution to the mechanism synthesis problem and depends on the initial input parameters – the location and orientation of the design objects.

Every pair of directed screw lines is processed in order to obtain the four vertices of the polygon that will represent the congruence plane in the environment, and to calculate coordinates of the normal to the plane and the vertices of the plane's generator line. Since the congruences are infinite planes, the size of their polygonal representations is set in the program at a predefined value that allows for relatively unobstructed display of multiple congruence planes. The congruence polygons and the generator lines are drawn using

OpenGL primitives, with fixed congruences rendered in red color and moving congruences rendered in yellow color.

If a congruence line other than the default generator line is being selected from either the fixed or the moving congruence planes, its location within a congruence plane is determined by projecting the point associated with the wand's location, **W,** perpendicularly onto the plane, resulting in point **W´** (Figure 4.2).

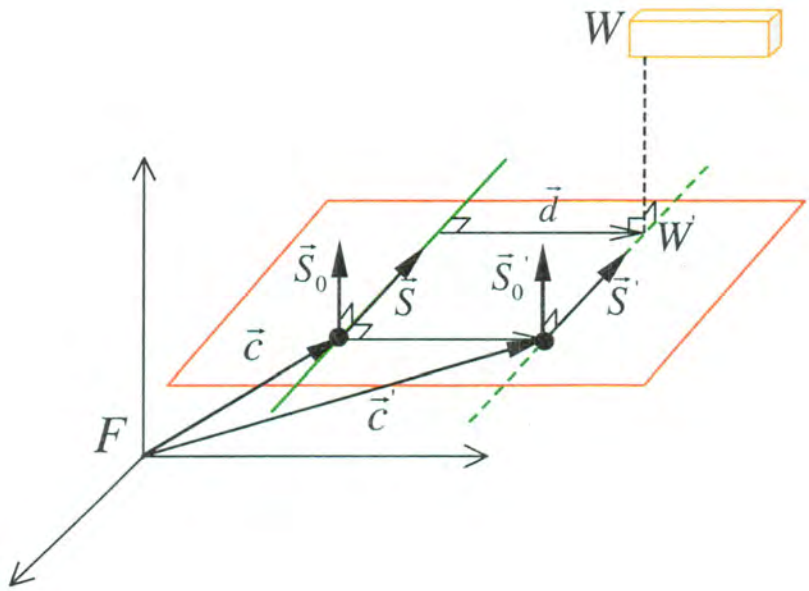


Figure 4.2: Selection of an Arbitrary Congruence Line.

Displacement vector $\vec{d}$, normal to the default generator line, is computed and used to determine a new vector $\vec{c}'$. To complete the new congruence line's definition, its unit directional vector $\vec{S}'$ is set equal to the unit directional vector $\vec{S}$ of the plane's generator line, and its moment vector $\vec{S}_0'$ is computed using formula $\vec{S}_0' = \vec{c}' \times \vec{S}'$. Since there is a one-to-one correspondence between the fixed and the moving congruences, the second congruence

plane and, thus, the direction of the second axis of the CC dyad is known. The dyadic dimensional synthesis technique for three spatial positions is then used to find the location of the second congruence line in that plane (Larochelle and Vance, 2000).

4.4 Guide Map Generation and Interaction

In order to utilize the guide map in the design process, the fixed and moving congruences are calculated as usual. Every possible combination of the driving and driven dyads, associated with the computed congruences, is then analyzed and the type of the corresponding mechanism is determined. The mechanism type information is stored in a data matrix. For N congruence planes, the size of the matrix is N x N. A second N x N data matrix, which identifies whether the mechanism has passed the circuit and branch defect filters, is computed at this point as well. The guide map is drawn as a square of a predefined size consisting of N x N square polygons, rendered next to each other using OpenGL primitives. The color of each polygon is determined by the type of the corresponding mechanism stored in the matrix. The guide map key identifying mechanism types according to color is created using GLF text and OpenGL primitives and displayed next to the guide map for user reference. As previously described, the selection from the guide map is done with a laser-like pointer extending from the wand. If the wand's button is released while the pointer is intersecting the guide map, the point of intersection is computed by projecting the point associated with the wand's position with respect to the coordinate system of the guide map onto the plane of the guide map itself along the pointer direction. The mechanism, associated with the selected location on the guide map, is computed and displayed in the environment.

4.5 Combined Interaction with the Guide Map and the Congruences

Any point on a guide map uniquely defines a set of fixed and moving axes for the two CC dyads of the mechanism. Following selection of a point on the guide map, its index identifications in the guide map data array can be determined. The two indices correspond to the location of the desired axes data for the driving and driven dyad within the fixed and moving congruence data arrays. This information is then retrieved and used to generate the linkage. To combine the display of the guide map with the congruences, the indices are used to locate the corresponding congruence planes in the VR environment and the color of the corresponding generator lines is changed.

Similarly, if a congruence plane is selected, its index is used to identify either the column or the row in the guide map data array corresponding to either the driving or the driven dyad selection. All data points contained in the identified column or the row will be drawn red on the guide map, resulting in a vertical or a horizontal line. This approach is also used for mechanism type identification and filtering options in the congruence planes. Following a single selection from the congruences, all data points contained in the associated column or row of the guide map data array are checked for the mechanism type identifier and the filtering flag. This information is then used to assign appropriate mechanism type colors to the congruence planes in the VR environment.

4.6 Speech Interface

Speech recognition is defined as the ability of electronic systems, such as computers, to identify the sound of a human voice and to use the voice messages to control the system

(Weinschenk and Barker, 2000). The ultimate goal of any virtual reality system is to provide the highest level of immersion to the users of a synthetic environment. This necessitates the ability to interact with the objects in the environment in addition to simply observing them. One way to accomplish this is with natural physical actions, such as grabbing and moving an object using a position-tracked input device. This method is an example of a direct manipulation interface (McGlashan and Axling, 1996). In order to further improve the 'naturalness' of the interaction process, it can be expanded to include human language. The combination of a speech interface with the ability to interact directly with objects in the virtual environment results in a multimodal interface where users can issue either physical (i.e. wand motion) or speech commands (McGlashan and Axling, 1996).

A speech interface in a virtual environment can be implemented using two different approaches: a fully interactive speech interface and a 'command and control' speech interface. A fully interactive speech interface has the ability to recognize a wide variety of words and phrases and can react to user's verbal actions in a highly flexible manner, resembling human-to-human dialog. These systems are generally quite adept at dealing with natural human speech and correctly identifying user's requests, even if they are uttered in an ambiguous manner. Unfortunately, these systems are usually speaker-dependent, and, therefore, each user must provide samples of his/her voice in order to train the system. This process is called enrollment (Noyes, 1993). While speaker-dependent software results in high recognition accuracy and extensive vocabulary, it lacks flexibility, since it cannot be easily shared among users. In situations where only a small set of predefined commands is required, command and control systems provide a better alternative, since they allow for the use of speaker-

independent methods. Speaker-independent systems do not rely on the enrollment process to tailor the application to a specific individual and can be used by multiple users. Generally, these systems are suitable for applications that require only a relatively small vocabulary of approximately 40 words or less, plus the digits 0-9 (Weinschenk and Barker, 2000). The set of interaction commands in the VRSpatial application made it an appropriate candidate for augmentation with a command and control voice interface.

4.6.1 VRSpatial speech interface components.

The C6 virtual environment is controlled by Silicon Graphics, Inc. (SGI™) computers running the IRIX® operating system. On the other hand, most of the commercially available speech-recognition software is designed to be used either on Microsoft® Windows® or Linux operating systems. Due to these considerations, two main tasks were identified: creation of a 'command and control' speech-recognition application on a Windows® computer and development of a communication method between the Windows® computer and an SGI™ system that executes the main VR application. The former was implemented with IBM ViaVoice™ for Windows® Release 8 Professional Edition and Speech for Java from IBM alphaWorks. The omniORB2 version 2.8 of the CORBA standard was used for communication purposes.

Speech for Java is a Java programming tool for incorporating IBM's ViaVoice™ speech technology into custom user interfaces. ViaVoice™ is a commercial speech recognition and synthesis program that can be used to control standard Windows® programs such as Microsoft® Word and Excel. In order to use ViaVoice™ as a speech engine for

another program, an API such as Speech for Java™ is needed. Speech for Java is an implementation of version 1.0 of the Java™ Speech Application Programming Interface (JSAPI) developed by Sun Microsystems, Inc. JSAPI specifies a cross-platform interface to support command and control recognizers, dictation systems and speech synthesizers (Sun Microsystems, 2002).

An essential part of a speech application is the grammar object. Within the grammar object are definitions of speech patterns and organization of speech that will be used in the application. Speech for Java relies on the Java™ Speech Grammar Format (JSGF), which is a platform-independent, vendor-independent textual representation of one type of grammar, a *rule grammar* (also known as a *command and control* grammar or *regular grammar*), for use in speech recognition. If speech synthesis is required in order to provide feedback to the user to confirm commands, the computer response is defined with the Java™ Speech API Markup Language (JSML). JSML is a text format used to annotate text input to speech synthesizers. It provides detailed information on how to speak text through definition of elements that control important speech parameters, such as pronunciations of words, emphasis and speaking rate (Sun Microsystems, 2002). Using the JSML, the quality, naturalness and understandability of synthesized speech output can be controlled.

OmniORB2 is an Object Request Broker (ORB) that implements the 2.3 specifications of the Common Object Request Broker Architecture or CORBA (Bolton, 2002). It uses Remote Procedure Calls (RPC) technology that allows an application to make a remote procedure call with the same amount of effort as making a local function call. The calling application is designated as a client and the called application is designated as a server. The

remote operations are grouped in interfaces, similar to C++ classes and are called CORBA objects. One of the most important benefits of CORBA is the location transparency that it provides. This means that operations on the CORBA objects are always invoked using the same syntax, no matter where on the network the CORBA objects are located. CORBA also offers programming language neutrality – both the client and the server code can be written in any of the supported programming languages (C, C++, Java, etc.). The interface to a CORBA object is defined using the Object Management Group (OMG) interface definition language (IDL). OMG is a declarative language that is passed through an IDL compiler to map the IDL file to a specific language for the client and the server sides (Brose et al., 2001).

4.6.2 VRSpatial speech interface implementation.

A dedicated Windows® 2000 computer with access to the local network is used to handle the operation of the speech recognition interface. Network access is necessary in order to communicate with the SGI™ computers that run the main VR application. To allow users maximum mobility in the C6 virtual environment a wireless microphone is used to provide the audio input to the computer.

The speech recognition interface is written entirely in Java. It uses Speech for Java routines in order to access ViaVoice's speech recognizer and speech synthesizer engines (Figure 4.3).

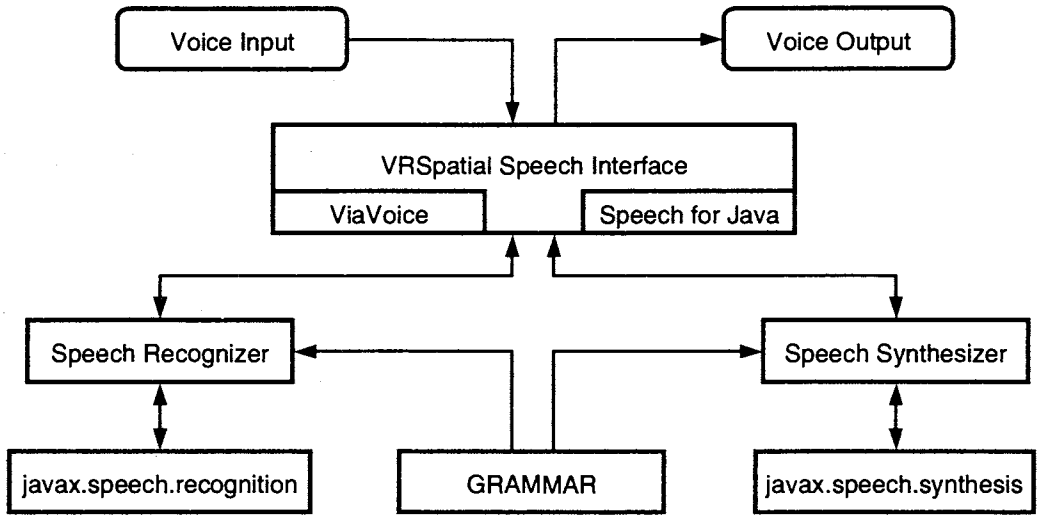


Figure 4.3: Speech Interface Structure.

The program continuously monitors the audio input by comparing it to the valid command patterns defined in the grammar file. The grammar file contains all of the valid action references ("Open", "Run", etc.) as well as many miscellaneous references that a user could possibly use while addressing the application ("Computer", "Please", etc.). As a way to provide maximum flexibility in the interaction process and to increase the range of supported user responses, several wording alternatives are also provided for a certain task (i.e. "Open Linkage", "Open Linkage File", "Load Linkage", etc.). A valid command must contain at least one action reference and an arbitrary number of miscellaneous references. Appendix B presents the contents of the VRSpatial grammar file.

If a valid command pattern is recognized, it is scanned for the presence of tags, which are associated with each action reference. Tags were used in order to simplify the processing of recognition results, since a single tag can be assigned to several instances of the same

action reference. For example, "Load" and "Open" action references are assigned the {open} tag, and similarly, "Start" and "Begin" are assigned the {start} tag. The usage of tags is not required by JSGF specifications, but it was found to be convenient in this particular case. The combination of tags is analyzed and a decision is made regarding the particular command that the user has issued. That command is then relayed to the VR application via the network. If the interface determines that the user has made an effort to issue a command, but no matching command patterns were found, then the speech synthesizer generates an appropriate message. The message, along with other possible verbal computer responses, is defined in a dedicated file that follows the JSML specifications.

To implement the networking capabilities, OMG IDL was used to define CORBA objects. An IDL compiler for Java was used to produce Java stub code. The stub code is used by the client (the speech interface in this case) to issue commands to the interface defined in the IDL file (VRSpatial in this case). Similarly, an IDL compiler for C++ was used to produce the C++ skeleton code, which is used by the server (VRSpatial) for definition of the CORBA object implementation.

CHAPTER 5. CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

Spatial mechanisms offer a better alternative to the traditional electronically controlled multiple-input devices, such as robotic manipulators, because they are more reliable and cost-efficient. Mechanism designers are forced to use traditional human-computer interfaces while synthesizing mechanisms for spatial tasks. The difficulty involved in the specification of the design problem in three-dimensional space and subsequent evaluation of the synthesized mechanisms necessitates development of novel computer applications. Such applications should allow mechanism designers to investigate the design environment and to define the design task in a more natural way compared to the traditional HCI. Virtual reality allows the users to interact with real size 3D models in a very natural way and it is well suited for the spatial mechanism design task. VRSpatial application was developed as a tool for designing spatial 4C mechanisms in virtual reality.

With the aid of VRSpatial, mechanism designers can synthesize spatial 4C mechanisms for rigid-body guidance through three or four precision positions. Utilizing the "design in context" approach, geometrical models of the design objects are manipulated in the VR environment in order to define the design problem. Fixed and moving congruences and the guide map are used to present the results of the position synthesis. Building upon the experience gained while operating the previous version of the spatial mechanism synthesis program, VRNETS, and following suggestions of its creators, the application developed as a result of this research has produced the next generation VR spatial mechanism design tool

which offers the designer an increased ability to effectively evaluate the design problem and its solutions.

In VRSpatial, with a switch to the VRJuggler programming platform, the range of VR systems capable of running the mechanism design program has been extended to include practically all of the modern VR hardware and software configurations. During the problem definition the user has gained the ability to perform numerical adjustment of the design positions and to manipulate the objects in a more intuitive way. New algorithms are used to filter the solutions that result in mechanisms with branch and circuit defects. A much wider variety of ways for presenting the solutions to the users have been implemented. Dynamic congruence computation, the ability to pick arbitrary lines within congruence planes, filtering and mechanism type identification in the congruence planes, and simultaneous interaction with the congruence planes and the guide map, provide more opportunities to obtain an optimal solution. Functionality for synthesis of mechanisms for rigid-body guidance through three precision positions accommodates design problems where four positions are not required. Furthermore, once a mechanism is synthesized, its behavior can be investigated in greater depth with the aid of extensive animation functionality. Finally, the level of interactivity within the application has been increased through the implementation of a speech recognition interface.

The result of this research is an advanced, multi-functional, and highly flexible application for design of spatial 4C mechanisms for motion generation tasks. It is hoped that the use of this program will facilitate expansion of the spatial mechanism investigation activities and will lead to their expanded use in industrial applications.

5.2 Recommendations for Future Work

Implementation of the three-position synthesis functionality in VRSpatial provides greater flexibility while designing mechanisms for such rigid-body guidance tasks where only the first and the last design positions are critical. The next step towards de-emphasizing the interaction with the intermediate positions is the two-position synthesis. Research work that is being done at the Florida Institute of Technology can serve as the basis for such functionality in VRSpatial.

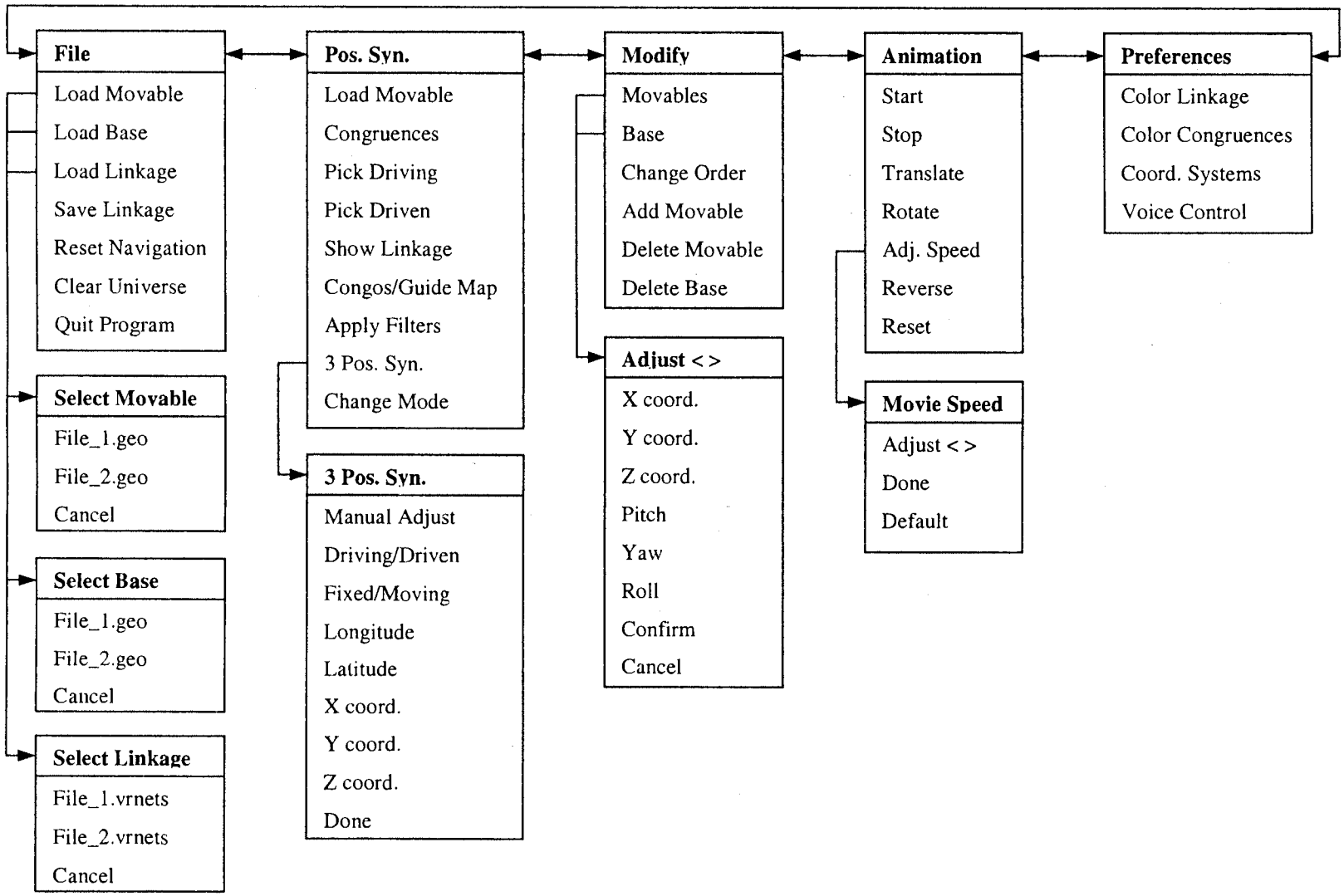
Currently, users rely exclusively on visual inspection in order to determine whether the moving objects in the design environment collide with each other or with the static surroundings when the synthesized mechanism is animated. Collision detection algorithms can be implemented in VRSpatial to detect the collision situations and respond by either identifying the colliding parts with visual cues or by halting the motion of the mechanism.

Currently, VRSpatial draws the mechanism links as straight lines, which directly connect mechanism joints. In the future, an option that allows the mechanism designers to change the shapes of the links in order to create a mechanism that is more suitable for production, or to use previously created CAD link models in the mechanism design process, could be provided.

Creation and operation of the software applications for mechanism synthesis, has led to the accumulation of significant experimental knowledge by the research teams at Iowa State University and Florida Institute of Technology. Research work contained in such software programs as VEMECS, Isis, VRNETS, and VRSpatial, as well as additional planar mechanism synthesis algorithms, could be combined in order to create a multi-functional

application for synthesis and analysis of planar 4R, spherical 4R, and spatial 4C mechanisms in virtual reality.

APPENDIX A. VRSPATIAL MENU STRUCTURE



APPENDIX B. VRSPATIAL GRAMMAR FILE

```
grammar hello;

public <comp_reference> = (C6 | computer | please ) *;

public <comp_reference_end> = (please | thanks | thank you | very much) *;

public <action> = Open {open} | Load {load} | Save {save} | Reset {reset} | Clear {clear}
                | Generate {generate} | Show {show} | Hide {hide} | Pick {pick}
                | Start {start} | Stop {stop} | Animate {animate} | Toggle {toggle}
                | Translate animate {translate} | Rotate animate {rotate} | Adjust {adjust}
                | Reverse {reverse} | Faster {faster} | Go faster {faster}
                | Play faster {faster} | Slower {slower} | Go slower {slower}
                | Play slower {slower};

public <required_obj> = (file {file} | movable {movable} | movables {movable}
                  | movable file {movable} | base {base} | base file {base}
                  | linkage{linkage} | linkages {linkage} | linkage file {linkage}
                  | menu {menu} | navigation {navigation} | universe {universe}
                  | congruence {congos} | congruences {congos} | congos {congos}
                  | congruence planes {congos} | guide map {map} | type map {map}
                  | map {map} | all {both} | both {both} | driving {driving}
                  | driven {driven} | filters {filters} | movie {animation}
                  | animation {animation} | RGB {RGB})*;

public <numbering_options> =[ one {one} | two {two} | three {three} | four {four}
                            | five {five} | six {six} | seven {seven} | eight {eight}
                            | nine {nine} | ten {ten}];

public <optional_obj> = [a | the | number | speed | color | colors];

public <command_skeleton> =
      <action><optional_obj><required_obj><optional_obj><numbering_options>;

public <UserCommand> = <comp_reference>{command_issued}<command_skeleton>
      <comp_reference_end>;
```

BIBLIOGRAPHY

Ascension Technology Corporation, 1993, *The Flock of Birds: Installation and Operation Guide*, Burlington, VT.

Barfield, W., Furness, T.A. III, 1995, *Virtual Environments and Advanced Interface Design*, Oxford University Press, New York, New York.

Bierbaum, A., Just, C., Hartling, P., Meinert, K., Baker, A., and Carolina Cruz-Neira, 2001, "VR Juggler: A Virtual Platform for Virtual Reality Application Development", *Proceedings of the IEEE Virtual Reality 2001 Conference (VR'01)*, Yokohama, Japan, March 2001.

Bolton, F., 2002, *Pure CORBA*, Sams Publishing, Indianapolis, Indiana.

Brose G., Vogel A., Duddy K., 2001, *Java Programming with CORBA. Advanced Techniques for Building Distributed Applications*, Wiley and Sons, New York, NY.

Bullinger, H.G., Breining, R., Bauer, W., 1999, "Virtual Prototyping – State of the Art in Product Design," *Proceedings of the 26th International Conference of Computers & Industrial Engineering,* Melbourne, pp. 103-107, December 15-17, 1999.

Chase, T.R., Mirth, J.A., 1993, "Circuits and branches of single-degree-of-freedom planar linkages", *ASME Journal of Mechanical Design*, 115:223-230, June 1993.

Chuang, J.C., Strong, R.T., Waldron, K.J., 1981, "Implementation of Solution Rectification Techniques in an Interactive Linkage Synthesis Program", *Journal of Mechanical Design*, 103: 657-664, July 1981.

Coats, B.A., Cipra, R.J., 1983 "A Computer Graphics Technique for Creating and Animating Spatial Linkages using IMP (Integrated Mechanism Program)", *Proceedings of 8th Applied Mechanisms Conference*, September 1983.

Deisinger, J., Breining R., Rößler, A., Höfle, J., Rückert, D., 2000, "Immersive Ergonomic Analyses of Console Elements in a Tractor Cabin," *4th Immersive Projection Technologies Workshop*, Ames, Iowa, June 19-20, 2000.

Engineering Animation Inc., 1998, *WorldToolKit User's Guide*, Mill Valley, CA.

Erdman, A.G., 1985, "Computer-Aided Design of Mechanism: 1984 and Beyond", *Mechanism and Machine Theory*, Vol.20, No.4, pp.245-249, 1985.

Erdman, A.G., Gustafson, J.E., 1977, "LINCAGES: Lincage, Interactive Computer Analysis and Graphically Enhanced Synthesis Package", *ASME Design Engineering Technical Conference,* Paper No. 77-DET-5, September 1977.

Evans, P.T., Vance, J.M., Dark, V.J., 1999, "Assessing the Effectiveness of Traditional and Virtual Reality Interfaces in Spherical Mechanism Design", *ASME Journal of Mechanical Design*, 121:507-514, December 1999.

Fakespace Inc., 1995, *Fakespace Pinch™ Glove System Installation Guide and User Handbook*, Menlo Park, CA.

Freudenstein, F., Sandor, G. N., "Synthesis of Path-Generating Mechanisms by Means of a Programmed Digital Computer", *Journal of Engineering for Industry*, 81(B):159-168, May 1959.

Furlong, T.J., Vance J.M., Larochelle, P.M., 1999, "Spherical Mechanism Synthesis in Virtual reality", *ASME Journal of Mechanical Design*, 121:515-520, December 1999.

Hartenberg, R. S. and Denavit, J., 1980, *Kinematic Synthesis of Linkages*, McGraw-Hill, New York, NY.

Katwyk, K.V., Cheng, H.H., 1997, "XLinkage: A Web-Based Analysis and Simulation Tool for Planar Mechanical Systems", *Proceedings of DETC'97: 1997 ASME Design Engineering Technical Conferences,* DETC97/DAC-3863, Sarcramento, CA, September 14-17, 1997.

Kemler, E.N., Howe, R. J., 1951, "Mechanism Design – punched-card method of constructing tables", *Machine Design*, 23: 148-152, September 1951.

Kihonge, J. N., Vance, J. M., Larochelle, P. M., 2001, "Spatial Mechanism Design in Virtual Reality with Networking", *Proceedings of the 2001 Design Engineering Technical Conference*, DETC2001/DAC-21136, Pittsburgh, PA, September 2001.

Kraal, J. C., 1996, "An application of virtual reality to engineering design: synthesis of spherical mechanisms", Master's Thesis, Iowa State University, Ames, IA, 1996.

Larochelle, P.M., Vance, J.M., 2000, "Interactive Visualization of the Line Congruences Associated with Four Finite Spatial Poses", *Proceedings of A Symposium Commemorating the Legacy, Works, and Life of Sir Robert Stawell Ball*, Cambridge, MA, July 2000.

Larochelle, P. M., 1998, "SPADES: Software for Synthesizing Spatial 4C Mechanisms", *Proceedings of DETC'98: 1998 ASME Design Engineering Technical Conferences,* DETC98/MECH-5889, Atlanta, GA, September 13-16, 1998.

Larochelle, P. M., 1995, "On the Design of Spatial 4C Mechanisms for Rigid-Body Guidance Through 4 Positions", *Proceedings of the 1995 ASME Design Engineering Technical Conferences*, Boston, MA, DE-82: 825-832.

Larochelle, P.M., Dooley, J.R., Murray, A.P., McCarthy, J.M., 1993, "SPHINX: Software for Synthesizing Spherical 4R Mechanisms", *NSF Design and Manufacturing Systems Conference*, 1:607-611, January 1993.

Mallik, A.K., Ghosh, A., Gunter Dittrich, 1994, *Kinematic Analysis and Synthesis of Mechanisms*, CRC Press, Inc., Boca Raton, FL.

McCarthy, J.M., 2000, *Geometric Design of Linkages*, Springer-Verlag, New York, NY.

McCarthy, J.M., 1995, "The Synthesis of Planar RR and Spatial CC Chains and the Equation of a Triangle," *Special Combined Issue of theASME Journal of Mechanical Design and Journal of Vibration and Acoustics*, 117(B):101-106, June 1995.

McCarthy, J.M., 1990, *An Introduction to Theoretical Kinematics*, The MIT Press, Cambridge, MA.

McGlashan, S., Axling, T., 1996, "A Speech Interface to Virtual Environments," *Proceedings of the International Workshop on Speech and Computers*, St. Petersburg, Russia.

Mechdyne Corporation, 2002, "MD Flex™", *Mechdyne Corporation,* http://www.mechdyne.com/pflex.shtml. Date accessed: July 3, 2002.

Murray, A., and McCarthy, J., 1994, "Five Position Synthesis of Spatial CC Dyads", *Proceedings of the 1994 ASME Design Engineering Technical Conferences*, Minneapolis, MN, DE-70:143-152, September 1994.

Myklebust, A., Keil, M.J., Reinholtz, C.F., 1985, "MECSYN-IMP-ANIMEC: Foundation for a New Computer-Aided Spatial Mechanism Design System", *Mechanism and Machine Theory*, 20:257-269, 1985.

Noyes J., 1993, "Speech Technology in the Future," In C. Baber & J. M. Noyes (eds) *Interactive Speech Technology: Human factors issues in the application of speech input/output to computers*, Taylor & Francis Ltd, London, pp. 189-208.

Oliveira, J.C., Shirmohammadi, S., Hosseini, M., Cordea, M., Georganas, N.D., Petriu, E., Petriu, D.C., 2000, "VIRTUAL THEATER for Industrial Training: A Collaborative Virtual Environment," *Proceedings of 4th World Multiconference on Circuits, Systems, Communications & Computers*, (CSCC 2000), Greece, July 2000.

Orlandea, N., Chace, M. A., 1977, "Simulation of a Vehicle Suspension with the ADAMS Computer program", SAE Paper. No. 770053, 1977.

Osborn, S. W., Vance, J. M., 1995, "A Virtual Reality Environment for Synthesizing Spherical Four-bar Mechanisms", *Proceedings of the 1995 Design Engineering Technical Conference*, Boston, MA. DE-83: 885-892, September 1995.

Podobedov, R., 2002, "GLF Library", *Romka Graphics,* http://romka.demonews.com/projects/glf/index_eng.htm. Date accessed: July 3, 2002.

Reinholtz, C.F., Sandor, G.N., Duffy, J., 1986, "Branching Analysis of Spherical RRRR and Spatial RCCC Mechanisms", *Journal of Mechanisms, Transmissions, and Automation in Design*, 108: 481-486, December 1986.

Rubel, A.J. and Kaufman, R.E., 1977, "KINSYN III: A New Human-Engineered System for Interactive Computer-Aided Design of Planar Linkages", *Journal of Engineering for Industry*, Vol. 99(B), No. 2, pp. 440-448, May 1977.

Ruth, D.A., McCarthy, J.M., 1997, "SphinxPC: An Implementation of Four Position Synthesis for Planar and Spherical 4R Linkages", *Proceedings of the 1997 ASME Design Engineering Technical Conferences*, September 1997.

Sheth, P. N., Uicker, J. J., 1972, "IMP (Integrated Mechanism Program), A Computer-Aided Design Analysis System for Mechanisms and Linkage",*Journal of Engineering for Industry*, 94(B): 454-464, May 1972.

Stuart, R., 2001, *The Design of Virtual Environments*, Barricade Books, Ft. Lee, NJ.

Suh, C.H., Radcliffe, C.W., 1978, *Kinematics and Mechanisms Design*, John Wiley & Sons, New York, NY.

Sun Microsystems, Inc., 2002, "Java™ Speech API Home Page", *Java.sun.com – The Source for Java™ Technology,* http://java.sun.com/products/java-media/speech. Date accessed: July 2, 2002.

Thatch, B.R., and Myklebust, A., 1988, "A PHIGS-Based Graphics Input Interface for Spatial Mechanism Design, *IEEE Computer Graphics and Applications*, 8:26-38, March 1988.

Tse, D.M., 2000, "Computer-Aided Synthesis of Mechanical Systems for Spatial Motion", Ph.D. Dissertation, Florida Institute of Technology, Melbourne, FL, 2000.

Vasiliu, A., Yannou, B., 2001, "Dimensional Synthesis of Planar Mechanisms Using Neural Networks: Application to Path Generator Linkages", *Mechanism and Machine Theory*, 36:299-310, February 2001.

Virtual Research Systems, Inc., 2002, "V8 Head Mount Display", *Virtual Research Systems, Inc.,* http://www.virtualresearch.com/products/v8.htm. Date accessed: July 3, 2002.

Wehage, R.A., Haug, E. J., 1982, "Generalized Coordinate Partitioning for Dimension Reduction in Analysis of Constrained Dynamic Systems",*Journal of Engineering for Industry*, 104: 247-255, 1982.

Weinschenk, S., Barker, D.T., 2000, *Designing Effective Speech Interfaces*, John Wiley and Sons, New York, NY.

Woo, M., Neider, J., Davis, T., Shreiner, D., 1999, *OpenGL® Programming Guide*, Addison-Wesley Co., Reading, MA.